REMARKS

Regarding paragraph 4 of the Office Action, the present amendment cancels reference numeral 633 from the specification. Reference numeral 406 is indeed present in Fig. 4 and has been added to the specification.

Regarding paragraph 5, reference numbers 200, 214, 236, 316, 604-618, 622-624, 700, 712-720, 1034, 1036, 1100 and 1120 have been added to the specification. Reference number 1052 is discussed in the first full paragraph of page 32 of the specification.

Regarding paragraph 6, the first use of each of the acronyms FAT, BIOS, ROM, PC, SCSI, FAT2, RAM, FAT16, FAT32 and NTFS has been spelled out. No instances of "QNK" were found.

No new matter has been added.

Each of claims 28, 30-36, 41-43, 82, 84 and 95-100 has either been canceled or amended so as not to include any trademarks or trade names. Reconsideration and withdrawal of the 35 USC §112(2) rejections are respectfully requested.

Claims 25-27, 29, 80-81 and 83 were rejected as being unpatentable over Helbig in view of England. Reconsideration and withdrawal of these rejections are respectfully requested.

Helbig et al.

On page 7 of the outstanding Office Action, the Office states that Helbig discloses:

> **providing and installing a trusted verification driver in the gaming machine, the trusted verification driver being independent of the operating system** (col. 8, 57-62, discloses a trusted operator independent of the operating system);

However, Helbig et al. do not teach any such trusted verification driver that is independent of the operating system, or any step of providing and installing such. In fact, Helbig et al.

specifically teach that the "trusted operator" that the Office analogizes to the claimed "trusted verification driver" (and its interface), in fact, runs under the control of the operating system – in this case the Disk operating System, or DOS:

> **To enable a trusted operator (TO) to specify which components of the SEPB's firmware and the PC's firmware and which files stored on one of the PC's disks will be protected (and at which step each protected component will be verified), a Trusted Operator Interface Program is provided with the SEPB. The Trusted Operator Interface Program, which runs under DOS on the PC and will, when executed, provide the trusted operator with a convenient mechanism to designate which information files (BIOS, interrupt table, DOS, autoexec.bat, config.sys, etc.) are to be signed with a Digital Signature (or some equivalent modification detection code) and when (at which of three steps indicated below) those signatures are to be verified. (Underlining added for emphasis)**

and

> **To provide a convenient method for a system administrator to request that the SEPB Sign a file, the Trusted Operator Program was developed to execute under DOS after all of the set-up steps have been performed. The Trusted Operator Program provides an interface to the trusted operator to determine which of nine possible operations the trusted operator wishes the SEPB to perform. After this information is entered the Trusted Operator Interface Program puts the data for the operation into known locations in the PC's hard disk and prepares a message in the standard format for the SEPB. The Trusted Operator Interface Program then calls the SEPB BIOS Extension program. The BIOS Extension program requests that the SEPB MYK-80 subsystem read the data from the PC's RAM and make it available to the SEPB I486 subsystem. The hidden parts of the SEPB BIOS Extension program and the private I486 and MYK-80 SEPB firmware then direct the SEPB through the appropriate steps to perform the requested operation. (Underlining added for emphasis)**

The Trusted Operator, therefore, cannot both "developed to execute under DOS" and be independent thereof, as urged by the Office. Quite simply, the Trusted Operator allows a system administrator, via the Trusted Operator Interface, to designate which files the SEPB is to digitally sign. The SEPB is the "Security Enhanced Processor Board" and "consists of a dual microprocessor arrangement of an Intel I486 CPU 24 and a RISC coprocessor MYK-80, 10,

programmed as a security coprocessor. The MYK-80 is a special purpose combination of an ARM6 RISC microprocessor, some amount of ROM to store the firmware that composes the routines, and other logic as needed to support the MYK-80 and external operations." Thus Helbig et al.'s system runs via a Trusted Operator that runs under the operating system to designate those files that are to be digitally signed by the security daughter board, the SEPB.

It is respectfully submitted that nowhere does Helbig teach or suggest that the Trusted Operator is "**independent of the operating system**", as required by the claims.

England et al.

It is noted that England is relied on for its teaching of "downloading at least one software module into the gaming machine; and a downloaded software module" (Office Action, Page 7). It is also noted that the only mention of "game" in England refers to input devices (See Column 6, lines 38-55). There are no mentions of "gaming machines" in England et al.

The England et al. reference teaches digital rights management operating systems (DRMOS). England et al. teach that the DRMOS is configured to "check itself" by loading a first layer of the Operating Systems (the boot loader, which loads a boot block of the OS), which then checks each of the remaining components of the OS to determine whether they have been signed by a trusted source:

> stage process. A securely booted computer runs a trusted program at startup. The trusted program loads an initial layer of the operating system and checks its integrity (by using a code signature or by other means) before allowing it to run. This layer will in turn load and check the succeeding layers. This proceeds all the way to loading trusted (signed) device drivers, and finally the trusted application(s).

This process is further detailed in England et al. at Column 11, beginning at line 39:

> Shortly after a computer is turned on or is reset, a small program called a boot loader is executed by the CPU (block **301**). The boot loader loads a boot block for a particular operating system. Code in the boot block then loads various drivers and other software components necessary for the operating system to function on the computer. The totality of the boot block and the loaded components make up the identity of the operating system.
>
> For a DRMOS, that identity can be trusted only if the boot block and the loaded components are trusted. In the embodiments described herein, all components are signed by a trusted source and provided with a rights manager certificate. An exemplary embodiment of the rights manager certificate is described below in conjunction with FIG. **9**.
>
> The operating system checks the signature of a component before loading it (block **303**). If the signature is valid (block **305**), the component has not been compromised by someone attempting to circumvent the boot process and the process proceeds to check the level of trust assigned to the component (block **307**). If the signature is not valid (or if there is no signature) but the component must be loaded (block **319**), the operating system will not assume the identity of a DRMOS upon completion of the boot process as explained further below.

According to England et al., the DRMOS is configured to prevent untrusted access to DRM-secured data and to "renounce" its once trusted identity (stored in an internal CPU register) if caused to load an untrusted software component, as taught at Col. 11, line 67 to Col. 12, line 7:

> the boot process. If the device requires the loading of an untrusted component after the boot process completes, a plug-and-play DRMOS must then "renounce" its trusted identity and terminate any executing trusted applications (block **323**) before loading the component. The determination that an untrusted component must be loaded can be based on a system configuration parameter or on instructions from the user of the computer.

England et al. teach various methods for determining whether an application can be trusted, and all such methods appear to rely on the DRMOS checking the signature of the software component against a list of trusted providers (see Col. 13, lines 19-35), by maintaining a boot log (see Col. 13, beginning at line 36) or by manipulating cryptographic key pairs (see Col. 13, beginning at line 60) and the like.

Much like the Helbig et al. reference, England et al. do not teach or suggest providing or installing a trusted verification driver that is independent of the operating system and using the trusted verification driver to perform a verification of the operating system and to check to code signature of downloaded software modules or components, as apparently acknowledged by the Office since it withdrew its previous §§102/103 rejections over England and added the Helbig et al. reference to the outstanding §103(a) rejection. As noted in an earlier response, in England et al., the verification of the OS is performed... by the OS itself (!), and not by a trusted verification driver that is independent of the operating system. The claimed methods and gaming machines of claims 24 and 80, therefore, are clearly antithetical to Helbig et al.'s "Trusted Operator" model that operates under the control of the DOS and also clearly antithetical to the OS-centric DRMOS model espoused by England et al., in which the OS checks itself.

The Helbig et al. and England et al. combination

The applied combination of references does not teach or suggest the claimed limitation ...

> **providing and installing a trusted verification driver in the gaming machine, the trusted verification driver being independent of the operating system;**

... as both references teach OS-centric security, wherein a Trusted Operator executing under DOS (Helbig et al.) or the operating system (DRMOS) itself carries out the security checks (England et al.).

The applied combination fails to teach or to suggest a trusted verification driver that is independent of the operating system or any steps of providing and installing the same, as claimed herein. In fact, the applied combination unambiguously teaches away from such a claimed step, as both teach that the entity that carries out the security measures is either a program that

operates under the operating system (DOS, Helbig et al.'s case), or is the operating system itself (in England et al.'s case).

That being established, the following claimed steps cannot be considered to be either taught or suggested by the applied combination:

> **performing a verification of components of the operating system against a trusted reference using the trusted verification driver and preventing further operation of the gaming machine when the verification of the components of the operating system fails;**
>
> **...**
>
> **checking a code signature of at least one downloaded software module using the trusted verification driver, and**
>
> **authorizing execution of the downloaded software module in the gaming machine only if the downloaded software module is successfully verified by the trusted verification driver**

This is because none of these steps are possible without the provision and installation of a trusted verification driver that is independent of the operating system.

That England teaches downloading content and, therefore, downloaded content, is not believed to remedy the fundamental shortcomings of the primary reference relied upon in the outstanding Office Action.

Claim 80 includes recitations that are similar to those of independent claim 24, and the arguments above are equally applicable thereto. For brevity's sake and rather than repeating them here, the remarks above are incorporated herein by reference, as if repeated here in full.

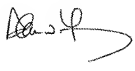
It is, therefore, respectfully submitted that the 35 USC §103(a) rejections of the claims are in error and should be withdrawn. The same is, therefore, respectfully requested.

Kindly note that all of the changes to the claims are related to the perceived 112(2) trademark and trade name issue and not to any art-related rejection. Therefore, any consideration that may be required is not believed to be such as could be reasonably characterized as being

"undue". Therefore, it is respectfully submitted that the present amendment is properly enterable after final rejection.

If any unresolved issues remain, the Examiner is respectfully invited to contact the undersigned attorney of record at the telephone number indicated below, and whatever is required will be done at once.

Respectfully submitted,



Date: August 6, 2010 By:

Alan W. Young
Attorney for Applicants
Registration No. 37,970

YOUNG LAW FIRM, P.C.
4370 Alpine Rd., Ste. 106
Portola Valley, CA 94028
Tel.: (650) 851-7210
Fax: (650) 851-7232